Page No.
Date

Module-1: Language Fundamentals

- Introduction
- Keywords
- Datatypes & Variables
- operators
- Control Statement
- Methods
- Identifiers
- Scanners
- Arrays
- Strings

Module-2: Object oriented Programming

- Introduction
- Static & Non-static
- Class & object
- Memory Management
- Initialization Blocks
- Constructors
- Inheritance
- Constructor chaining
- Typecasting
- Method overloading
- Method Overriding
- Polymorphism
- Abstraction
- Encapsulation

Page No.
Date

Module 3 : Core Java Libraries

→ Introduction
→ Class object
→ Exception Handling
→ Class String
→ StringBuffer & StringBuilder
→ Wrapper classes
→ Multithreading
→ File Handling
→ Collection Framework

- Software :- It is a collection of programs that works together to satisfy a requirement.
- Program :- A program uses set of inst. that performs a certain task.
- Syntax :- The structure of statements.
- Basic SYNTAX :- The Basic Syntax for writing Java a program Consists of two parts.

1) CLASS :- Class is the most basic fundamental unit of writing Java program.

2) Main Method :- The main method is an entry point for execution inside a Java program.

Page No.
Date

```
Class ClassName
Braces ⟶ {
    public static void main (String [ ] args)
                ↖ Parenthesis        ↖ Brackets
    {

    }
}
```

Public → Access Modifier
Static → Modifier
Void → return type
main → name of method
String → non-primitive
[ ] → Arrays
args → Arguments

```
Class FirstJavaProgram
{
  public static void main (String [ ] args)
  {
    System.out.println ("Welcome to Java");
  }
}
```

output :- Welcome to Java

Page No.
Date

* System.out.println();

→ It is a predefined statement in Java Syntax.
→ It is used by the programmer to display the Values to output screen
→ String type values should be Enclosed in Double quotes (" ").
→ Numbers can be displayed directly.
→ Single characters must be Enclosed in Single quotes (' ').
→ Boolean & logical values can be displayed directly.

```
Class Example
{
public static void main (String[] args)
{
System.out.println ("@spiders Jspiders");
System.out.println (143);
System.out.println (6.74);
System.out.println ('A');
System.out.println (true);
System.out.println (false);
}
}
```

output
@spiders Jspiders
143
6.74
A
True
False

* Write a Java program to display the below details
1. Full Name
2. YOP
3. CGPA
4. Gender
5. Single (T/F)

```
Class Myself
{
    public static void main (String[] args)
    {
        System.out.println ("Aman Raj");
        System.out.println (2024);
        System.out.println (7.2);
        System.out.println ('M');
        System.out.println (true);
    }
}
```

output:
Aman Raj
2024
7.2
M
true

* The "+" operator

→ In Java using the + operator we can perform the following two operations.

i) Addition

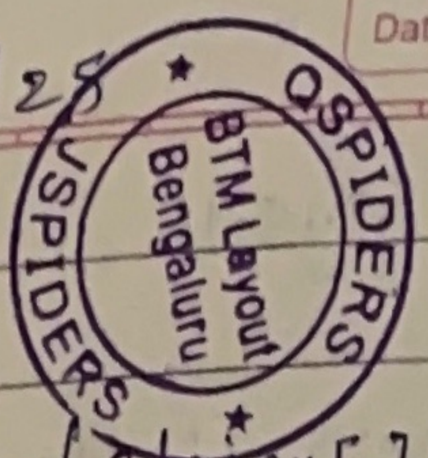
ii) Concatenation

→
```
Class Demo
{
    public static void main (string[] args)
    {
        System.out.println (10+10);
        System.out.println (10+2.3);
        System.out.println (3.2+10);
        System.out.println (2.3+3.2);
    }
}
```

output:
20
12.3
13.2
5.5

Page No.
Date

Java
13/2/25



→ Class Sample
{
public static void main (String[] args)
{
System.out.println("QSpiders" + "Jspiders");
System.out.println("Boys" + 90);
System.out.println(143 + "students");
}
}

output:
QSpiders Jspiders
Boys 90
143 students

→ Class Tricks
{
public static void main (String[] args)
{
System.out.println(10+10); → 20
System.out.println("10" + "10"); → "1010"
System.out.println("10" + 10); → "1010"
System.out.println(10 + "10"); → "1010"
System.out.println("A" + 10); → "A10"
System.out.println('A' + 10); → 75
}
}

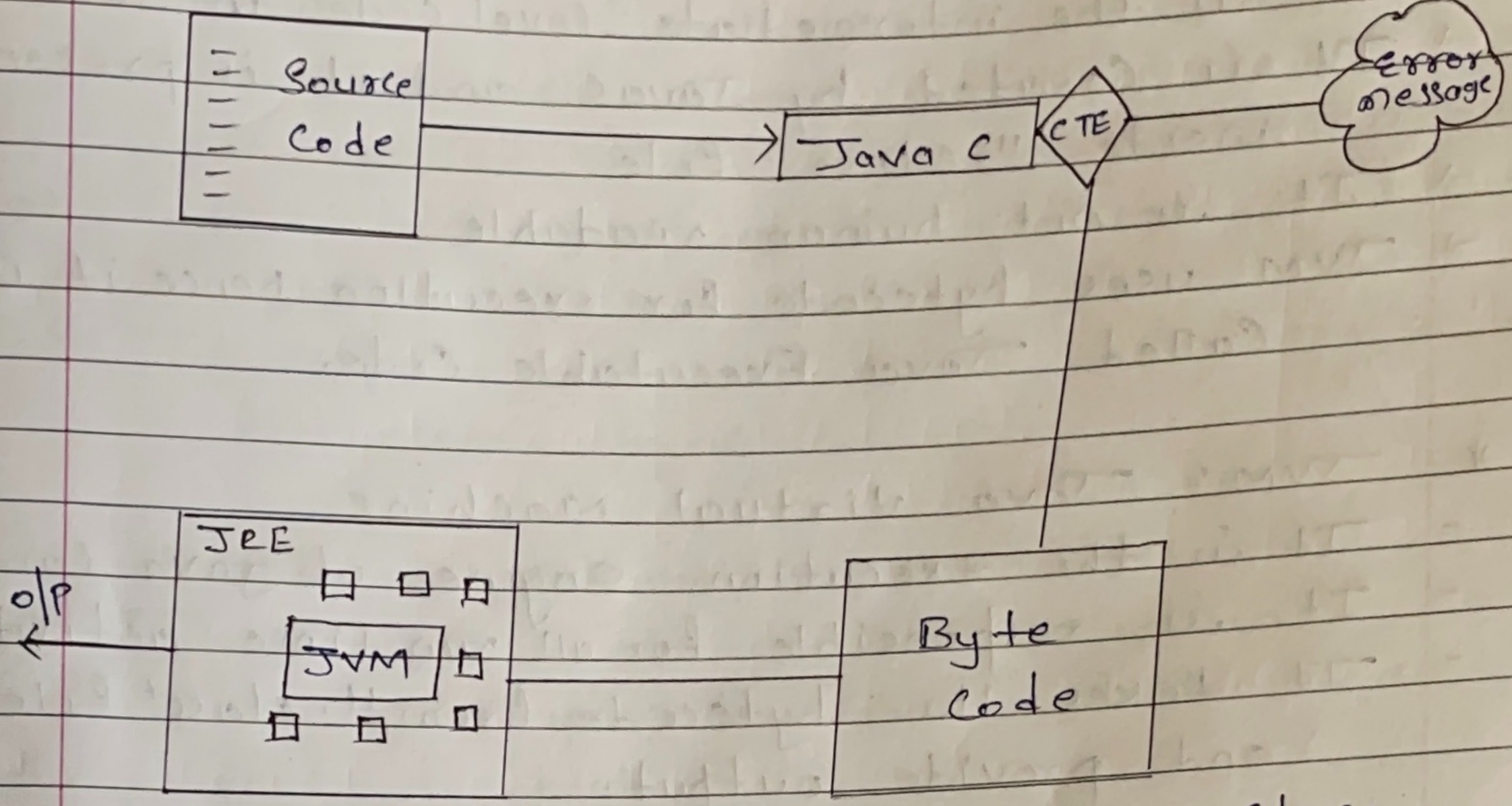
output:

Note :- Every character is internally represented using a number derived from UNICODE System in Java.

'A' — 'Z'		'a' — 'z'		'0' — '9'	
65	90	97	122	48	57

Page No.
Date

* Java Program LIFE CYCLE



- The programmer writes Javas Source Code and Saves it in Computer by providing a filename and "Java" file extension.

- The Source Code in ".Java" file given to Javac which checks for mistake, if mistakes are present it generates Error messages, if there are no mistakes it produces ".class" file which Contains bytecode.

- The Bytecode present in ".class" file is given to the JVM, the JVM executes the bytecode with the help its Subsystems in the JRE to provide output.

- Programmer performs basic output analysis.

* Bytecode:
→ It is the intermediate level code in Java
→ It is created by Javac and it is present inside ".class" file
→ It is not human readable
→ JVM uses bytecode for execution hence it is called Java Executable Code.

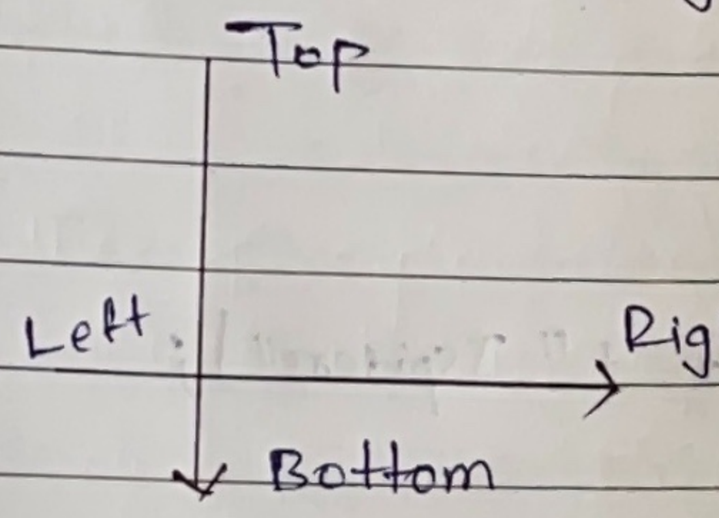
* JVM: Java Virtual Machine
- It is the execution Engine of Java lang
- It is responsible for all run time activities
- It takes the bytecode from ".class" file and provide output.

* JRE: Java Runtime Environment
- During program execution JVM internally makes use of several tools like
  Loader
  Linker
  Verifier
  Managers
  Event handlers
  etc.....
- This collection of tools used by the JVM is known as the JRE
- All activities inside the JRE is Controlled by the JVM.

Page No.
Date

* Javac
→ Javac is the name of Java Compiler
→ It takes the Source Code present in ".Java" file as an input & checks for mistakes.
→ It mistakes are found it displays appropriate error messages.
→ If mistakes are not found it generates ".class" file with bytecode.

* Order of Execution.
→ The default order of execution in a Java program is top to bottom and left to Right.



* COMMENTS: are written by the programmer in Java Source Code. description about the Code functionality.

→ In Java Comments are of two types
  i) Single line Comments
  ii) Multiline Comments.

→ Comments make the program more Easy to Read and Understand
→ Comments do not have any impact on program Execution.

```
/*
aim: to demonstrate the use of Comments
author: Anvesh
release: QSP-JSP-btm-All
*/
Class Example
{
/*
main() method:
It is the Energy point for execution inside Java program
*/
Public static void main(String[] args)
{
    // Addition
System.out.println(12+13);
    // Concatenation
System.out.println("Java "+"JSpiders");
}
}
```

* KeyWords
- KeyWords are inbuilt Words that have a special meaning
- We can use KeyWords But We Cannot Change the meaning.
- KeyWords are Completely in lowercase
- Generally KeyWords appear in a Shade of Blue Color.
- Java provides a total of 53 KeyWords
- byte, short, int, long, float, double, char, boolean, void, class, enum, interface, extends, implements. if, else, Switch, case, break, default, for, do, While

assert, Continue, static, final, abstract, Synchronized, Strictfp, transient, native, volatile, return, new, instanceof, this, super, true, false, null, public, private, protected, package, import, try catch, finally, throws, throw

goto Const

Note:,

goto & Const keywords are not use directly in Java programming.

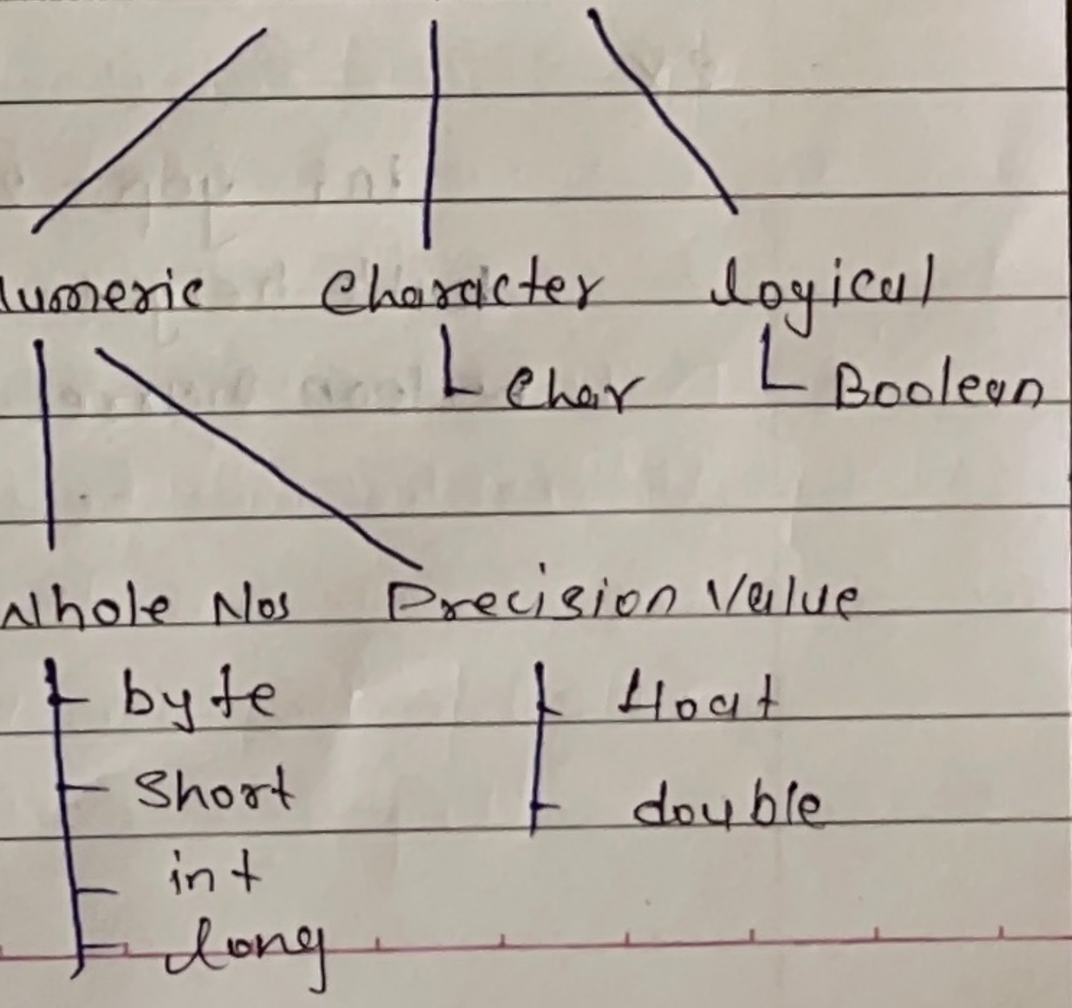
* Datatypes & Variables.

Datatypes:- are tokens in a Java program that helps to identify the type of values.

- In Java datatypes are classified into 2 types:-
  - i) primitive
  - ii) Non- primitive

Primitives:- are predefined data types
- Primitives are used to declare Variables
- Java provides a total of 8 primitives.

Byte - 1 Bytes
Short - 2 Bytes
int - 4 Bytes
long - 8 Bytes
float - 4 Bytes
double - 8 Bytes
char - 2 Bytes
boolean - 1 Bit



Page No.
Date

Note:- String is not a primitive datatype in Java.

Variables:- A variable is a Container that is used to Store a value
- The Variable name is always decided by the programmer

Syntax: Approach 1

datatype VariableName; → Declaration
VariableName = Value; → Initialization

Ex:-

```
int age;
age = 16;
double Cgpa;
Cgpa = 6.3;
Char ch;
ch = 'J';
```

Approach 2

datatype VariableName = Value;

Ex

```
int yop = 2010;
double height = 5.4;
boolean married = true;
```

Page No.
Date

* Storing values of type float & long

All float values should be suffixed with 'f' or 'F'

```
float num = 2.3F
float val = 3.4F
```

All long type values should be suffixed with 'l' or 'L'

```
long aadhar = 115678928876L;
long phoneno = 9608420121L;
```

* Re-initialization :- Replacing the old value in a variable with a new variable is called Re-initialization

```
int a; → Declaration
a = 100; → Initialization
:
:
a = 500; → Re-Initialization
```

E.g →

```
int ml;
ml = 30;
S.o.pln(ml); → 30

ml = 60; // re-initialization
S.o.pln(ml); → 60

ml = 90; // re-initialization
S.o.pln(ml); → 90
```

```
char ch;
ch = 'a';
ch = 'b';
ch = 'c';

Sopln(ch); → c
Sopln(ch); → c
Sopln(ch); → c
```

Page No.
Date

Note:

```
int x = 15;
So.p(x);
int x = 25;
```
⟶ CTE : re-declaration

```
int x = 15;
S.o.p(x);
double x = 2.5;
```
→ CTE : Re-declaration

During Re-initialization programmer should not provide Data-type.

* Final Variables : Any Variable that is declared with Final Keyword is Called as Final Variable.

```
final int a = 10;
final double b = 7.4;
final char c = 'J';
```

Final Variables Cannot be Re-initialized they are used to Create Constant Values.

```
int x = 15;
S.o.p(x);
x = 30;
S.o.p(x);
```

```
final int x = 15;
S.o.p(x);
x = 30;
```

* Data Members & Local Variables
- Variables can be declare Directly in the class body or inside a method.
- If We declare Variables directly in class body they are Called as Datamembers

- If We declare variable inside the method they are called Local Variables.

```
Class Sample
{
    int a = 10;
    double b = 4.5;
    Char c = 'J';
```

— Data Members.

```
Public static void Main (String[] args)
{
    x = 15;
    y = 7.8;
    z = 'Q';
```

— Local Variables.

```
}
}
```

* OPERATORS : are Symbols that performs a particular task.

- In Java are classified into following families :-

↳ Arithmetic operators

+, -, *, /, %, =

↳ Relational operators

<, >, <=, >=, ==, !=

↳ Logical operators

&&, ||, !

↳ Bitwise operators

&, |, ^, <<, >>

↳ Unary operators

++, --

Page No.
Date

Arithmetic operators

+ ——→ Addition ~~operator~~/ Concatenation
- ——→ Subtraction
* ——→ Multiplication
/ ——→ Division (quotient)
% ——→ Modulus (reminder)
= ——→ Assignment

* Biggest Datatype

i) Ability to store Precision
ii) Memory Size

(float / double) → double (int / int) → int
(int / long) → long (double / double) → double
(float / long) → float (int / double) → "
(long / double) → double (double / int) → "

* operator Precedence

[*, /, %] → 1st
[+, -] → 2nd
[=] → 3rd

* Relational oPerators

< — Less than
> — Greater than
<= — Less Than or Equal To
>= — Greater Than or Equal To
== — Equals To
!= — Not Equals To

NOTE:-

- Relational operators also known as Conditional operators are used for Comparison
- After the Comparison it returns boolean value True or False

* Logical operators

&& & —— Logical AND
|| —— Logical OR
! —— Logical NOT

Logical operator are generally used to ~~pick~~ build Complex Conditions that involved multiple Comparisons.
- Logical operators will Compare boolean values and provide answer in boolean format.

Decimal → Binary

2	90	= 1011010	2	25	= 11001
2	45 - 0		2	12 - 1	
2	22 - 1		2	6 - 0	
2	11 - 0		2	3 - 0	
2	5 - 1			1 - 1	
2	2 - 1				
	1 - 0				

Binary → Decimal

1 0 1 1 0 1 0

$1\times2^7+0\times2^6+1\times2^5+1\times2^4+0\times2^3+1\times2^2+0\times2$

$128+0+32+16+0+4+0$

$= 179$

* Bitwise operator

Symbol	Operation
&	BITWISE AND
|	BITWISE OR
^	BITWISE XOR
<<	BITWISE LEFT SHIFT
>>	BITWISE RIGHT SHIFT

BITWISE AND (&)

Input 1	Input 2	Input 3
0	0	0
0	1	0
1	0	0
1	1	1

BITWISE OR (|)

Input 1	Input 2	Input 3
0	0	0
0	1	1
0	1	1
1	0	1
1	1	1

BITWISE XOR (^)

Input 1	Input 2	Input 3
0	0	0
0	1	1
1	0	1
1	1	0

```
int p=11;
int q=13;
S.o p ln (p & q); 9
S.o p ln (p | q); 15
S.o p ln (p ^ q); 6
```

1011 AND
1101
1001
8+0+0+1 = 9

1011 OR
1101
1111
8+4+2+1 = 15

1011
1101
0110
0+4+2+0 = 6

32 16 8 4 2 1

Page No.
Date

49
7

```
int i = 23;
int j = 50;
S.o.pln(i&j);   18
So.pln(i|j);    55
So.pln(i^j);    37
```

```
0 1 0 1 1 1   —&
1 1 0 0 1 0
-------------
0 1 0 0 1 0  -> 18
x x x x x x
32 16 8 4 2 1
```

```
0 1 0 1 1 1   —|
1 1 0 0 1 0
-------------
1 1 0 1 1 1  -> 55
x x x x x x
32 16 8 4 2 1
```

```
0 1 0 1 1 1   —^
1 1 0 0 1 0
-------------
1 0 0 1 0 1  -> 37
32 16 8 4 2 1
```

* `int q = 9;`
`S.o.pln(q<<2);` -> 36

```
1 0 0 1
1 0 0 1 0
1 0 0 1 0 0
x x x x x x
32 16 8 4 2 1
-------------
36
```

* `int S = 5;`
`S.o.pln(S<<3);` -> 20

```
0 1 0 1
0 1 0 1 0
0 1 0 1 0 0
32 16 8 4 2 1 = 20
```

* BITWISE RIGHT SHIFT

`int c = 13;`
`So.pln(c>>1);` —6

```
1 1 0 1
1 1 0  = 6.
```

`int q = 40;`
`So.pln(q>>2);` 10.

```
1 0 1 0 0 0
1 0 1 0 0
1 0 1 0   = 10
```

4
32/16
8/4

`int s = 420;`
`So.pln(s>>>3);` 52

```
1 1 0 1 0 0 1 0 0
1 1 0 1 0 0 1 0
1 1 0 1 0 0 1  -> 105
1 1 0 1 0 0  -> 52
```

*

* Unary operator:- Java provides the following two unary operators
  i) ++ (increment)
  ii) -- (decrement)

i)- The increment operator will increase the value in a variable by 1

ii)- The decrement operator will decrease the value in a variable by 1.

Increment & Decrement operations can be done in two ways:-
  i) Post operation
  ii) Pre operation

Post operation:- In case of post operation first value in the variable will be used and then increment or decrement will be done

Syntax:- Variable ++
or
Variable --

Eg. int a = 12; //13, //14
int b = a++ ⊕ a++ ⊕ a++;
Sop ln (b); 39

int c = 18; //17, //16
int d = c-- ⊕ c-- ⊕ c--;
Sop ln (d); 51

Page No.
Date / /

Eg int x = 14; //13
int y = 16; //17
int z = x-- + y++ + x-- + y++;
14 16 13 17
S.o.println(z); -60.

Pre operation:- Incase of preoperation first increment or decrement is done & then value will be used.

Eg. int x = 17;
int y = 11
int z = --x + ++y + --x + ++y;
16 12 15 13
So println(z); -56

-> int x = 14; //13
int y = 18; //17
int z = x-- + y-- + ++x + --y;
14 18 14 17
So println(y);

* Control statement:- Java provides the following Control statements.

1) IF - statements
2) SWITCH- "
3) FOR. Loop
4) WHILE LOOP.

NOTE: The programmer can make use of Control statements to Alter the order of execution. inside a Java program.

IF-Statement :-

- Version 1: Simple IF statement

```
if (condition)  <  True
                   False.
{
  //it-body
}
```

- Version-2: IF-ELSE-Statement

```
if (condition)
{
  // it body
}
else
  //else body
}.
```

- Version-3: If-else-if- statement / ladder

```
if (condition-1)
{
}
else if (condition-2)
{
}
else if (condition-3)
{
}
!
:
else
{
}
```

Page No.
Date / /

* SWITCH - STATEMENT

Syntax:- Switch (choice)

```
{
    Case label1;
    break;
    Case label2;
    break;
    Case label3:
    break;
    Case label4;
    break;
    Case label5;
    break;
    :
    default:
}
```

FOR LOOP

```
for(initialization; condition; operation)
                ①          ②          ④
{
    ③ // for - loop body
}
```

e.g

```
for(int a=1 ; a<=5 ; a++)
{
    S.O.println("Biryani");
}
```

a = 1 Biryani
a = 2 Biryani
a = 3 Biryani
a = 4 Biryani
a = 5 Biryani
a = 6 ✗

Page No.
Date

Eg for(int x=5 ; x>0; x--)
{
S.o.pln("Bangloxe");
}

x=5 ✓ Banglore
x=4 ✓ "
x=3 ✓ "
x=2 ✓ "
x=1 ✓ "
x=0 x

Eg. for(int p=1; p<=10; p=p+2)
{
S.o.pln("Jspiders");
}

p=1 ✓ Jspiders
p=3 ✓ "
p=5 ✓ "
p=7 ✓ "
p=9 ✓ "
p=11 x

Eg: for(int q=10; q>0; q=q-3)
{
S.o.pln("Java");
}

q=10 Java
q=7 Java
q=4 Java
q=1 Java
q=-2 x